בס"ד

Generative Adversarial Network

Presenting:

Ben Remez

Barack Samuni

Omri Sason

Barak Yaakov

Generative Adversarial Network

- GAN - Explanation & Loss Function
- CycleGAN – Short Explanation
- Super Resolution
- SRresCycGAN – deep explanation, Arch, Loss, etc...
- The problem + Dataset
- Experiments + Metrics + Paper

Generative Adversarial Network

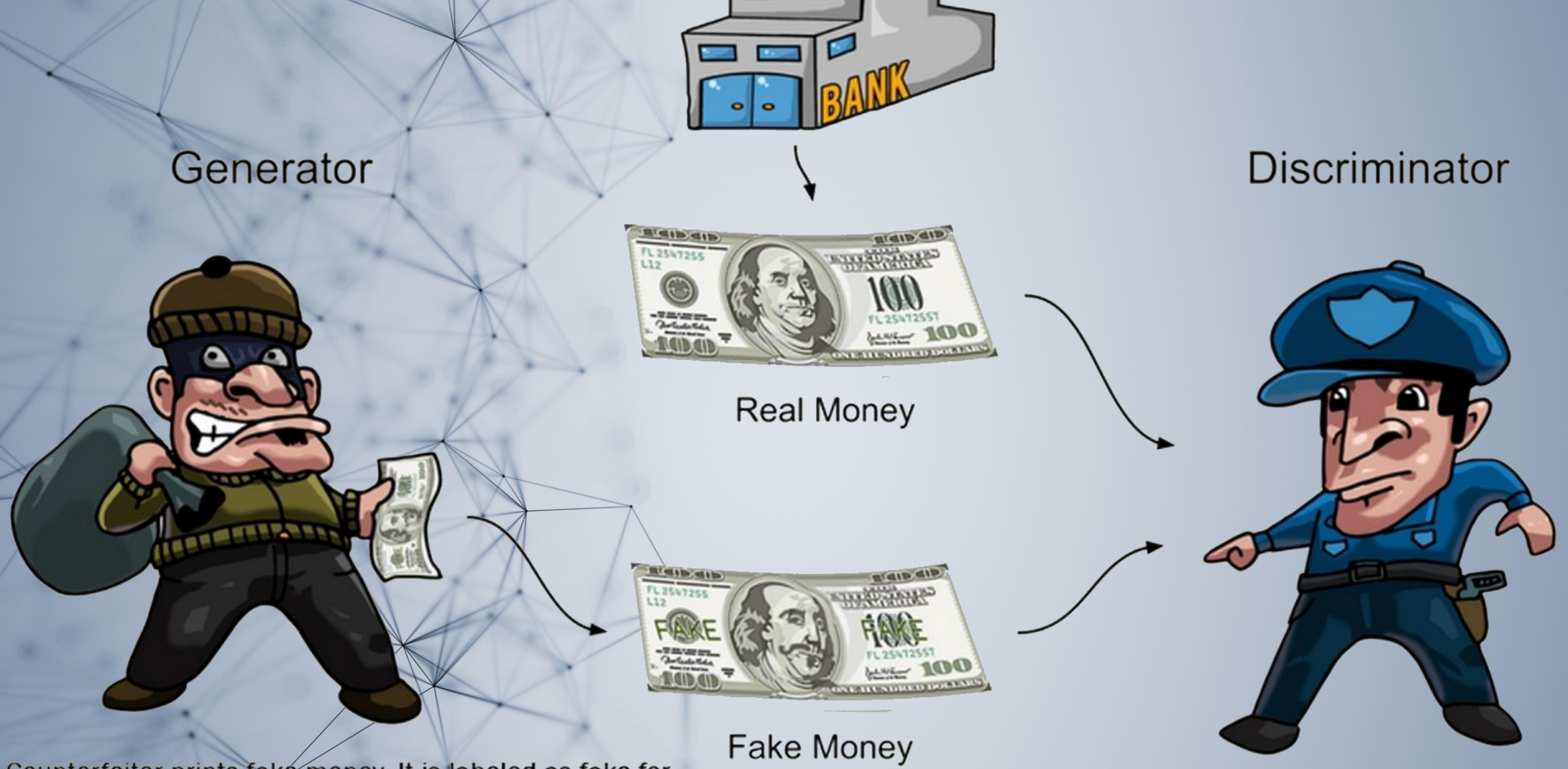
GAN

- Generative
- Adversarial
- Network

Generative Adversarial Network (GAN)

What is a Generative Adversarial Network (GAN)?

- Generative Adversarial Network (GAN) was designed by Ian Goodfellow and his colleagues in 2014.
- GANs are used to generate new, synthetic data that resembles a given training dataset.
- GANs are known for their ability to create realistic images, videos and audio.
- A GAN is built from two main components: A Generator and a Discriminator.



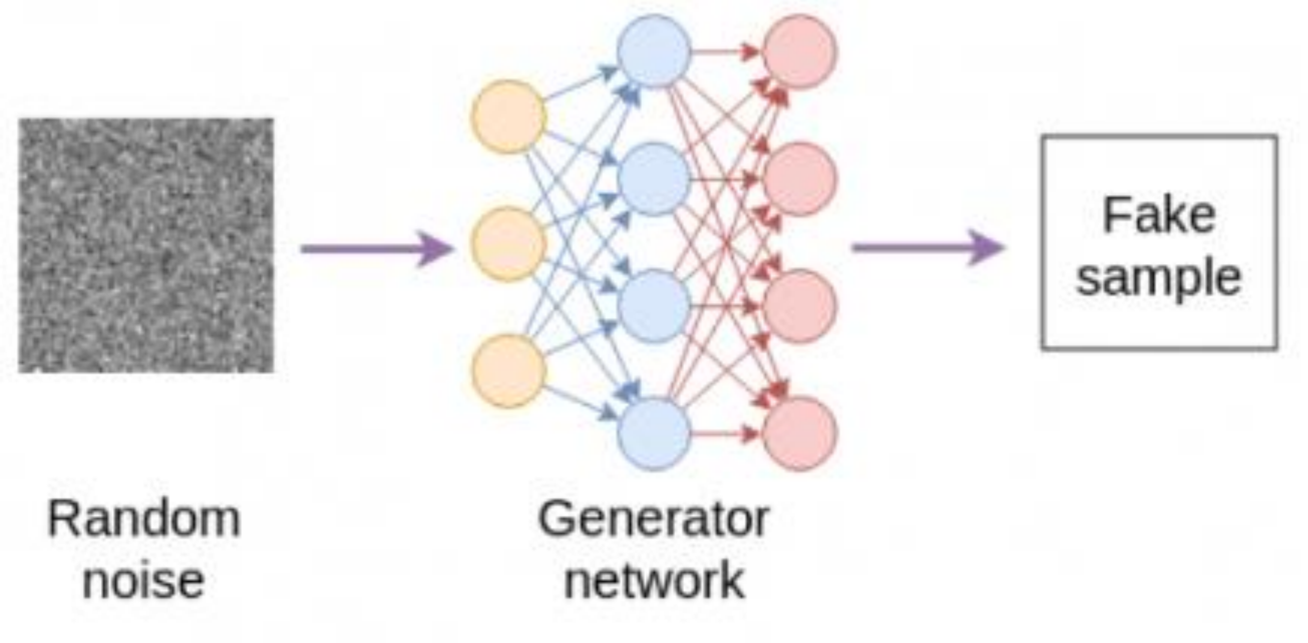
Counterfeiter prints fake money. It is labeled as fake for police training. Sometimes, the counterfeiter attempts to fool the police by labeling the fake money as real.

The police are trained to distinguish between. Sometimes, the police give feedback to the counterfeiter about why the money is fake.

Generative Adversarial Network

Generative

The GAN consists of a generator that aims to generate fake images as real-looking as possible.



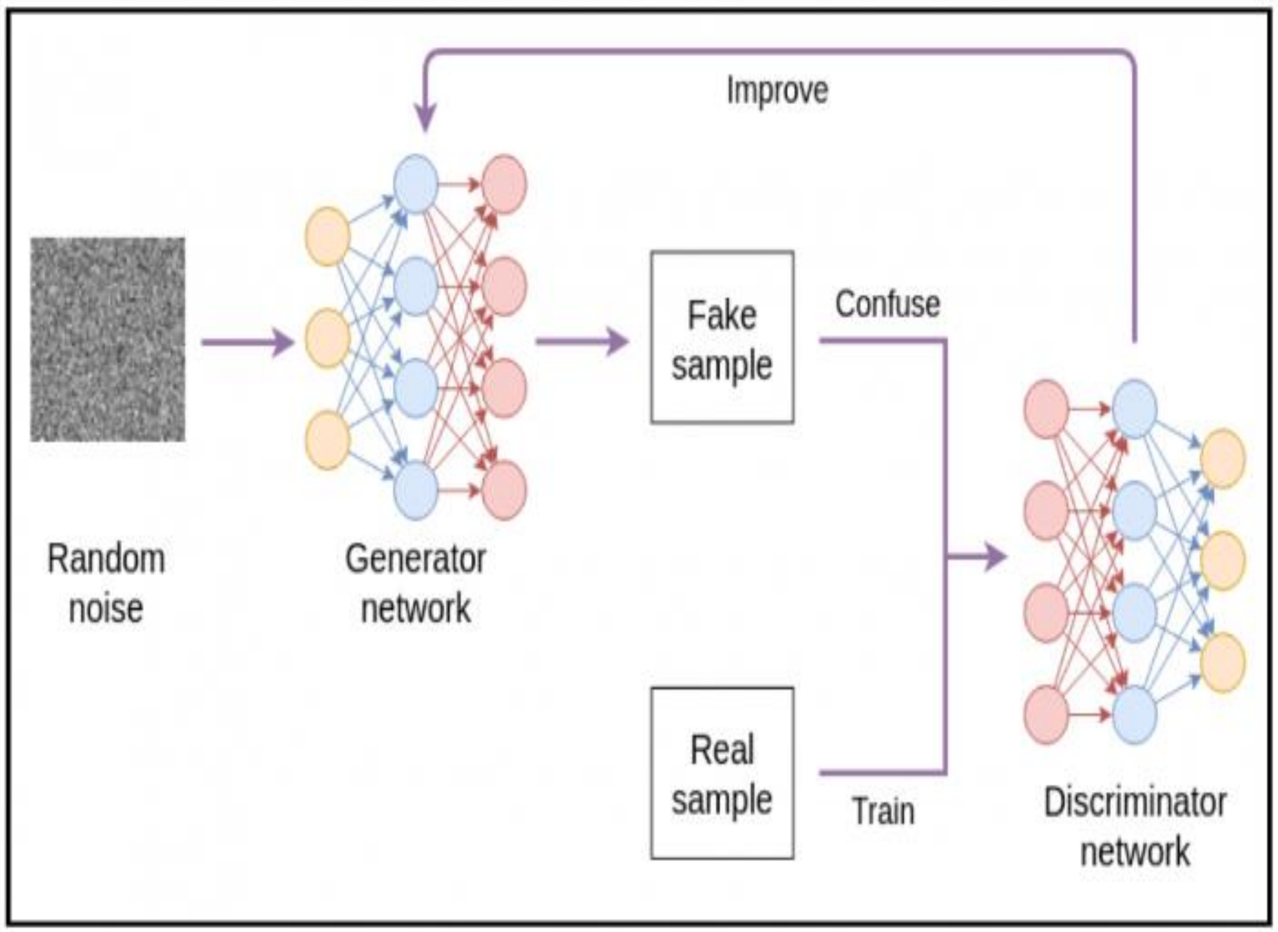
Generative Adversarial Network

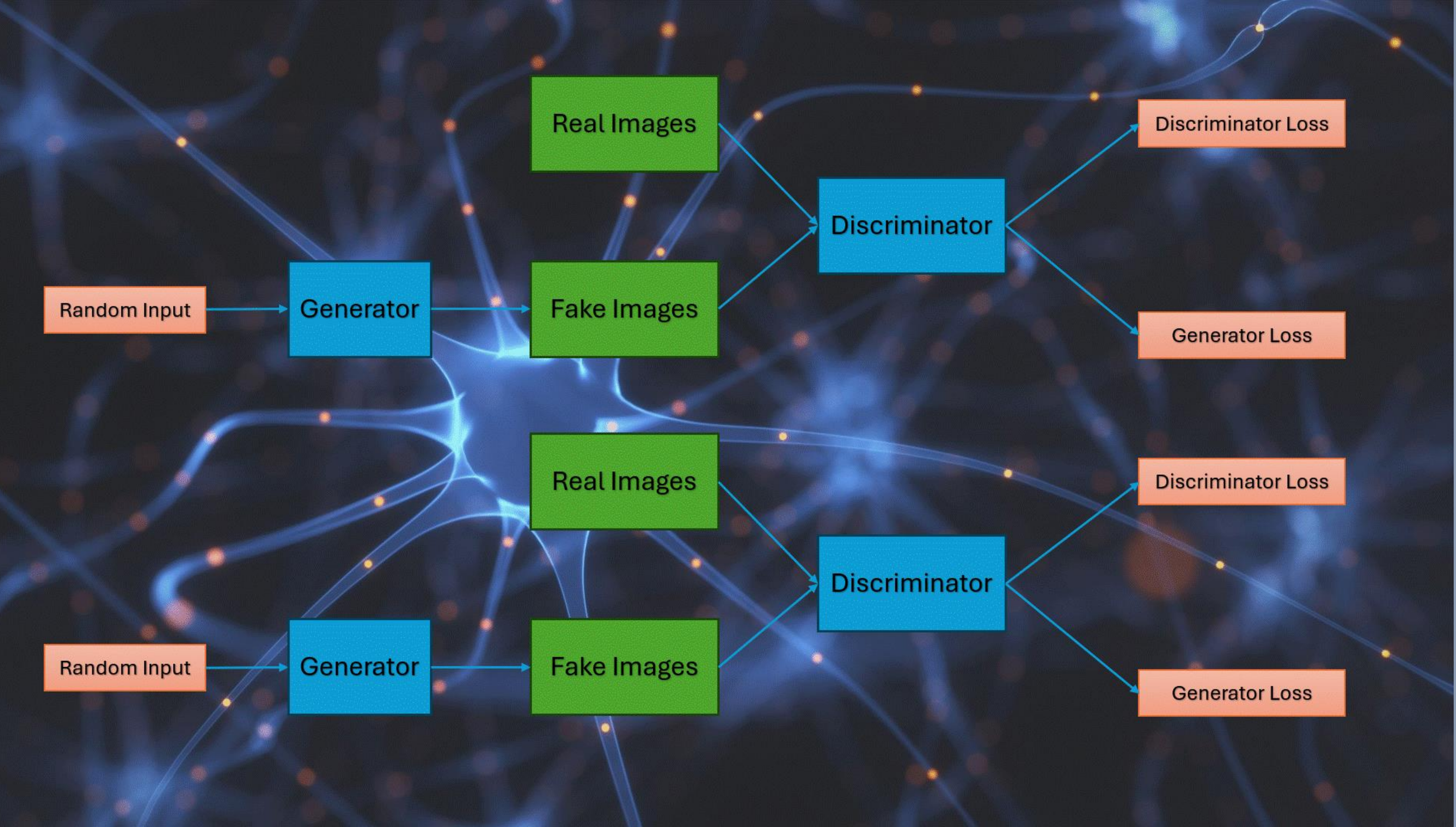
Adversarial

- Discriminator- The GAN also consists of a discriminator, that aims to distinguish between fake generated images and real ones.

- Generator- the generator trains to "fool" the Discriminator into thinking that his fake generated images are real ones.

- Meanwhile, the discriminator trains to be better in identifying the fake images.




Real Images
Discriminator Loss
Discriminator
Random Input
Generator
Fake Images
Generator Loss
Real Images
Discriminator Loss
Discriminator
Random Input
Generator
Fake Images
Generator Loss

Generative Adversarial Network

Applications

Image to Image Translation

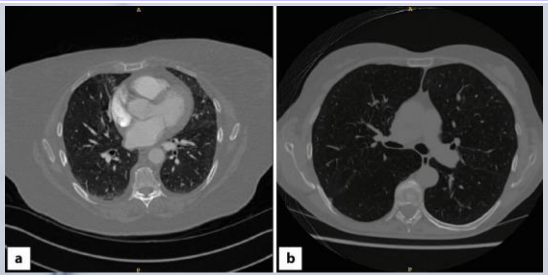
- Inpainting
- Super Resolution
- Colorization
- Image restoration

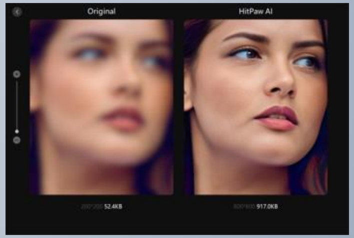
Data Synthesis

- Medical
- Autonomous Driving
- Art
- Video, Image, Text

Text to Image Generation

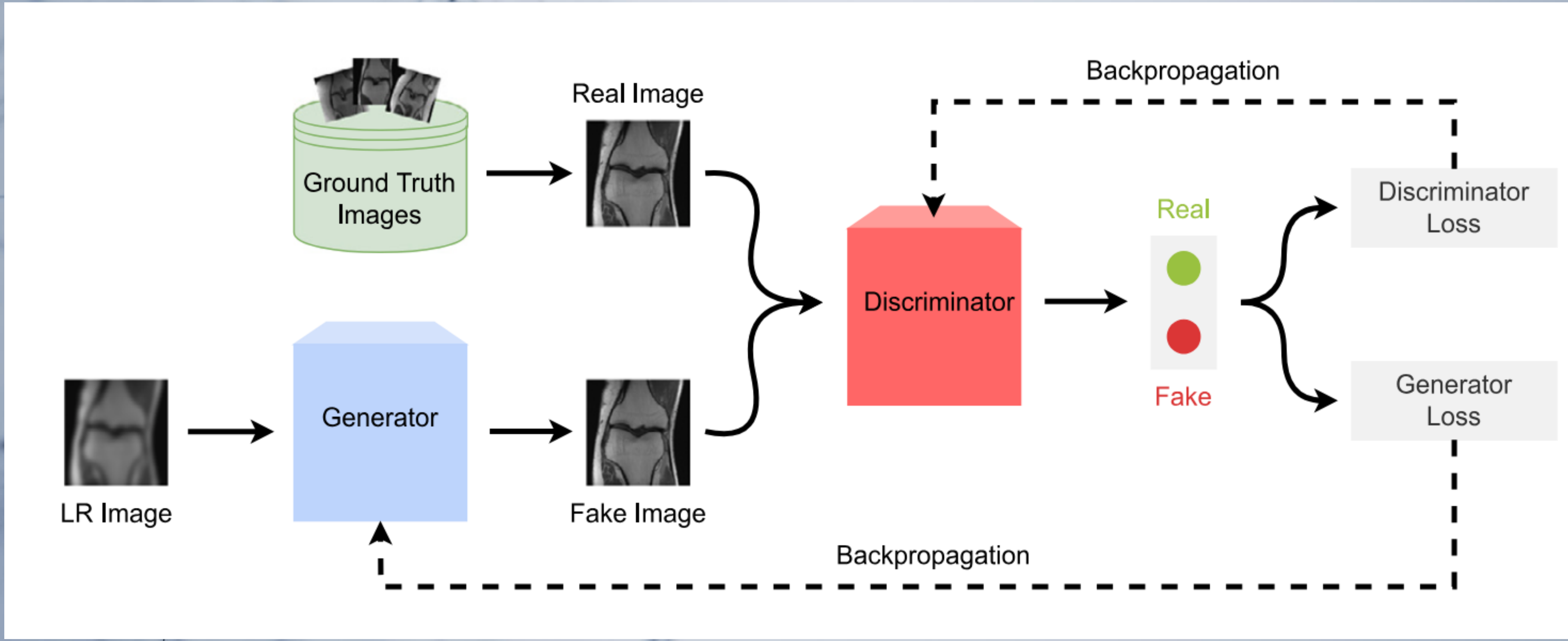
- E-commerce
- Visual Storytelling
- Healthcare





Generative Adversarial Network (GAN)

How do we TRAIN a Generative Adversarial Network?



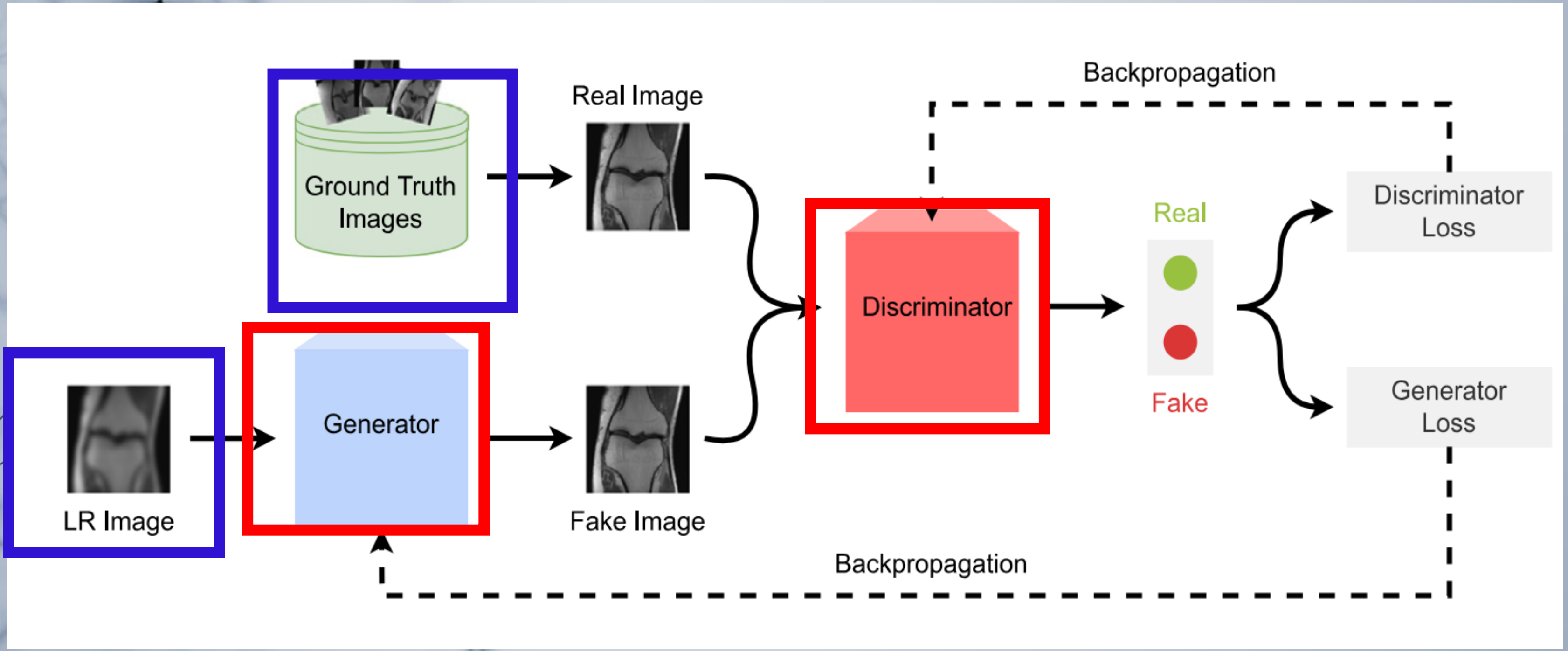
Adversarial Training Steps

Initialization:

- Initialize the generator and discriminator models.
- Set hyper parameters such as the number of epochs, batch size, learning rate and noise dimension.
- Define optimizers and the loss function.

Training Loop:

- Loop through the dataset for a specified number of epochs.
- For each batch in the training data:



Generative Training steps - Generator:

Training Loop Generator:

Generate new fake images using fresh noise.

Use labels of 1 (as the generator aims to fool the discriminator into thinking fake images are real).

Perform a forward pass through the discriminator using only fake images and compute the loss.

Perform backward pass and update the generator’s weights.

Adversarial Training steps:

Training Loop Discriminator:

- Get real images and create corresponding labels (1 for real).
- Generate fake images using the generator and create corresponding labels (0 for fake).
- Concatenate real and fake images and labels.
- Perform a forward pass through the discriminator and compute the loss.
- Perform backward pass and update the discriminator’s weights.

Ending the training:

- The training ends when the rate of the generator’s success to fool the discriminator passes a given limit (for example when the discriminator’s success rate is close to random choice).

Adversarial Loss for GAN

$$
\begin{aligned}
C(G) &= \max_D V(G, D) \\
&= \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}}[\log(1 - D_G^*(G(\boldsymbol{z})))] \\
&= \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x} \sim p_g}[\log(1 - D_G^*(\boldsymbol{x}))] \\
&= \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}(\boldsymbol{x})}{P_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right] + \mathbb{E}_{\boldsymbol{x} \sim p_g}\left[\log \frac{p_g(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right]
\end{aligned}
$$

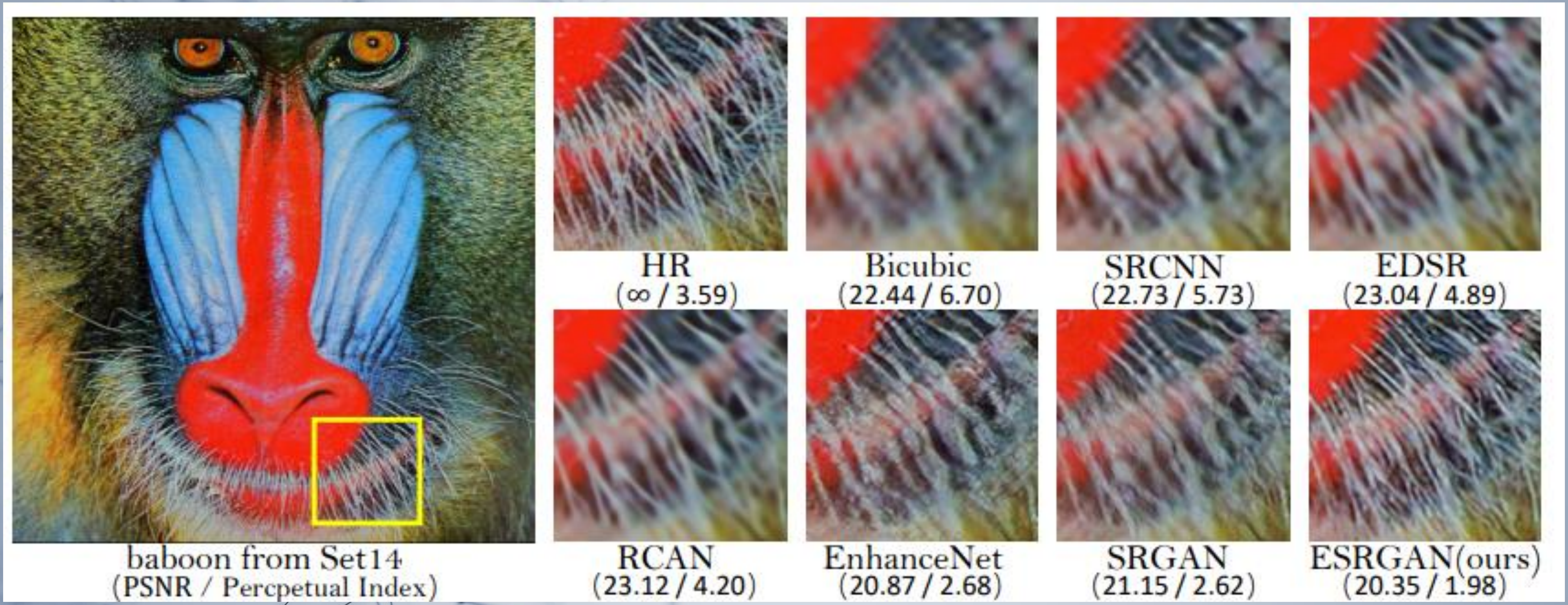
$$
\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].
$$

PROS & CONS of GAN for super-resolution

Pros:

- High Quality Results – Generates realistic and detailed high-resolution images
- Detail Enhancement – Adds fine textures, making images look natural.
- Learning Complex Distributions – Captures real-world nuances effectively.
- End-to-End Training – Directly maps low-resolution to high-resolution.
- Versatility - Applicable to various image types and domains.

PROS & CONS of GAN for SUPER Resolution



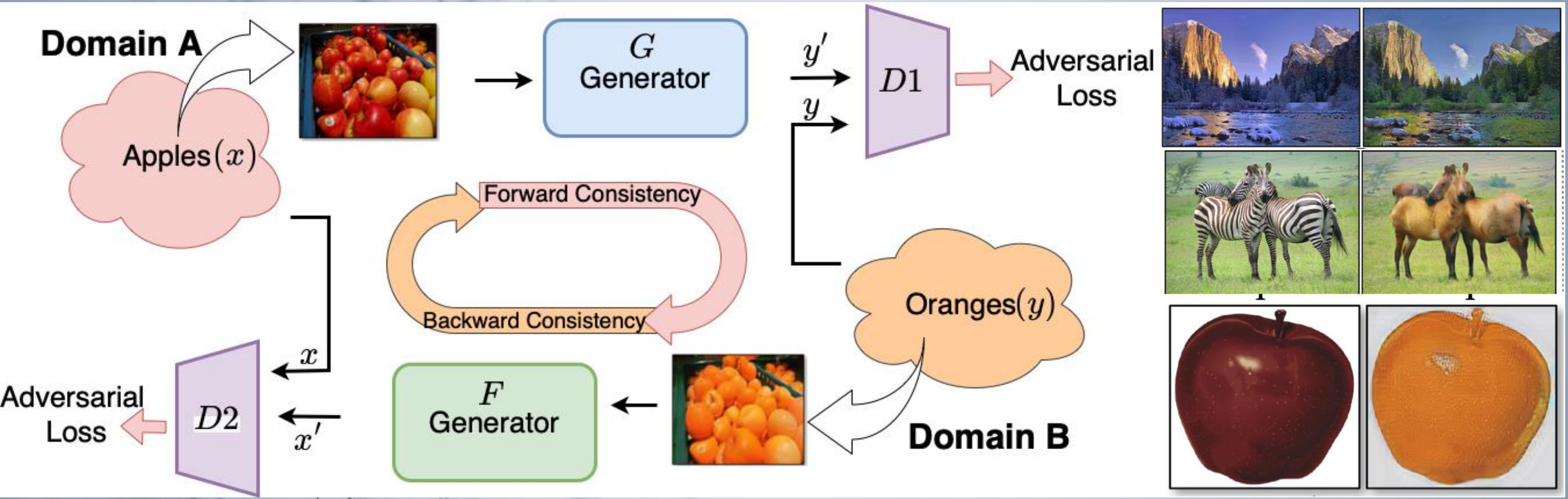
PROS & CONS of GAN for Super Resolution

Cons:

- Training instability – Requires careful tuning and is hard to train.
- Mode collapse - Can produce limited variety of images.
- Resource intensive – Needs significant computational power and time.
- Evaluation difficulty – Hard to measure perceptual quality improvements.
- Data dependency – Needs large amounts of high-quality training data.

Cycle GAN

Network architecture



Cycle GAN

CycleGAN uses cycle-consistency loss for training.

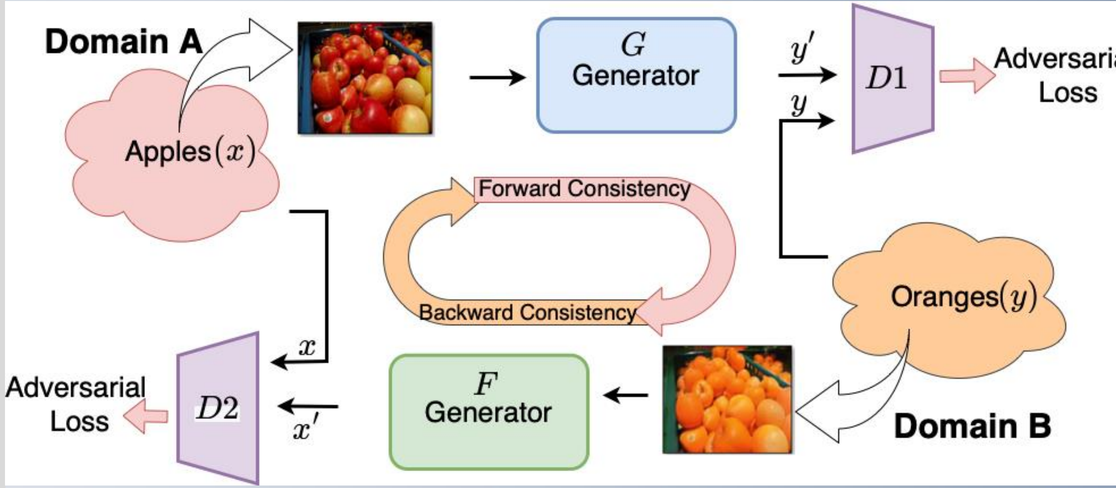
To ensure that the learned mappings **G** and **F** are consistent

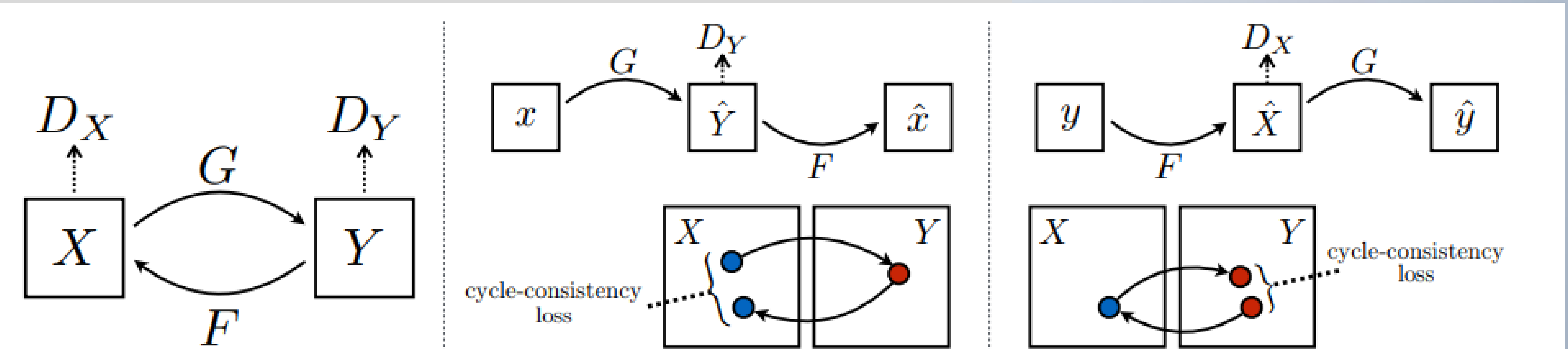
Generator G: Translates images from domain X to domain Y.

Discriminator D_Y: Differentiates between real images in domain Y and generated images G(X).

Generator F: Translates images from domain Y to domain X.

Discriminator D_X: Differentiates between real images in domain X and generated images F(Y)

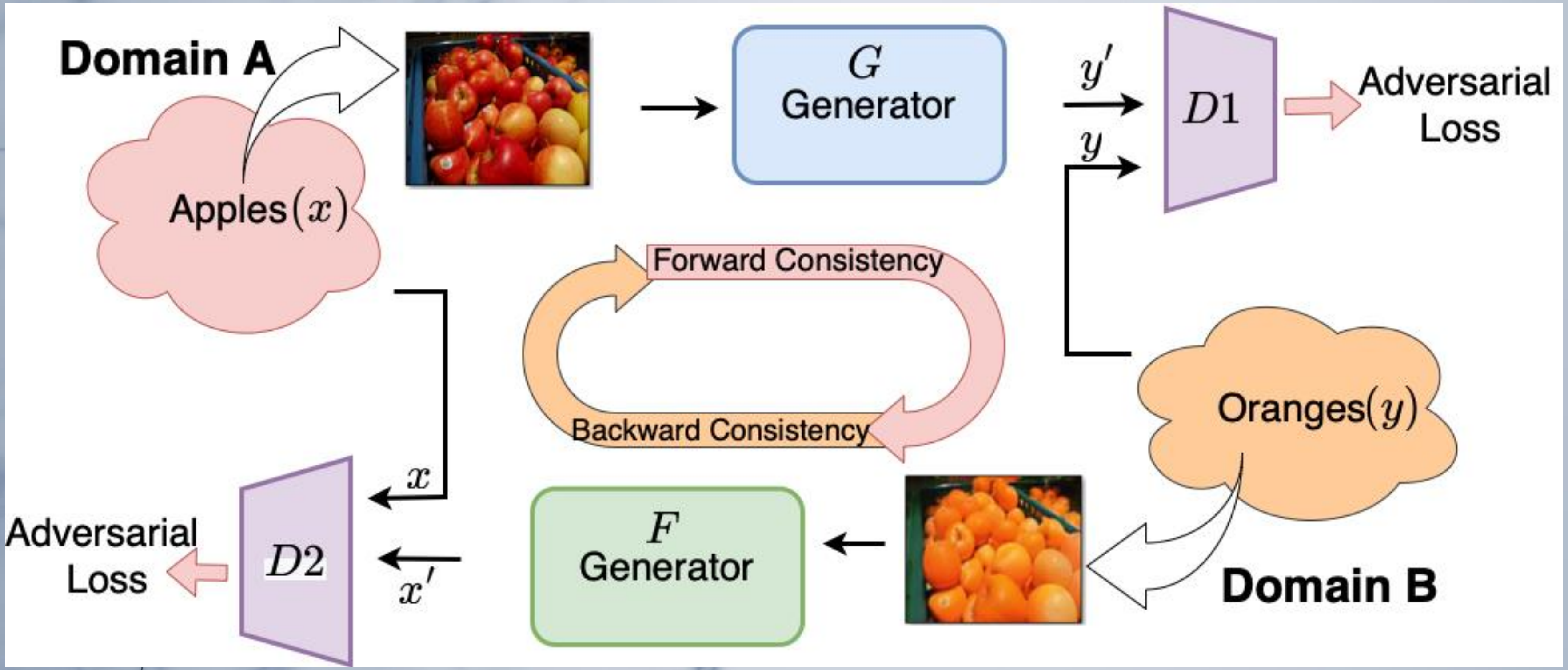




Cycle GAN Loss Functions

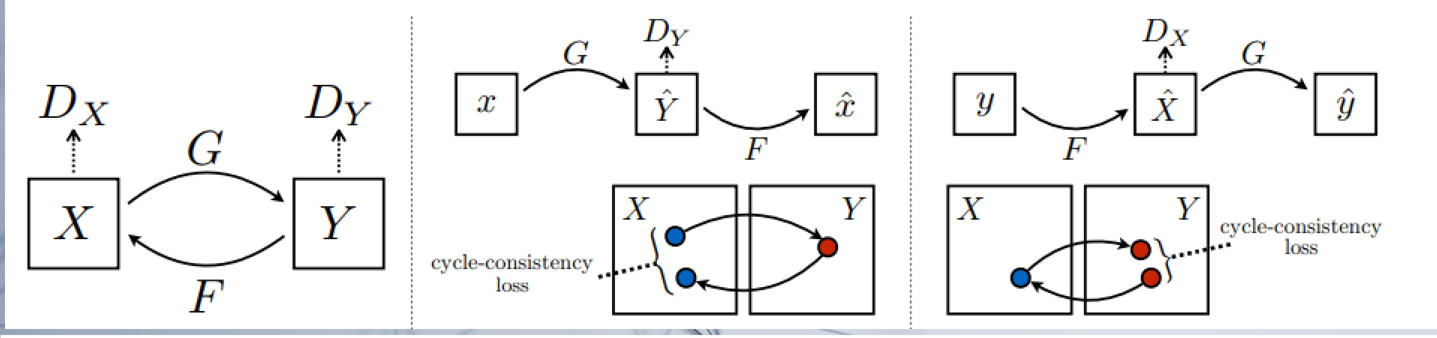
Training Optimization

$$G^*, F^* = \arg\min_{G,F} \max_{D_x, D_Y} \mathcal{L}(G, F, D_X, D_Y).$$



Cycle GAN Loss Functions

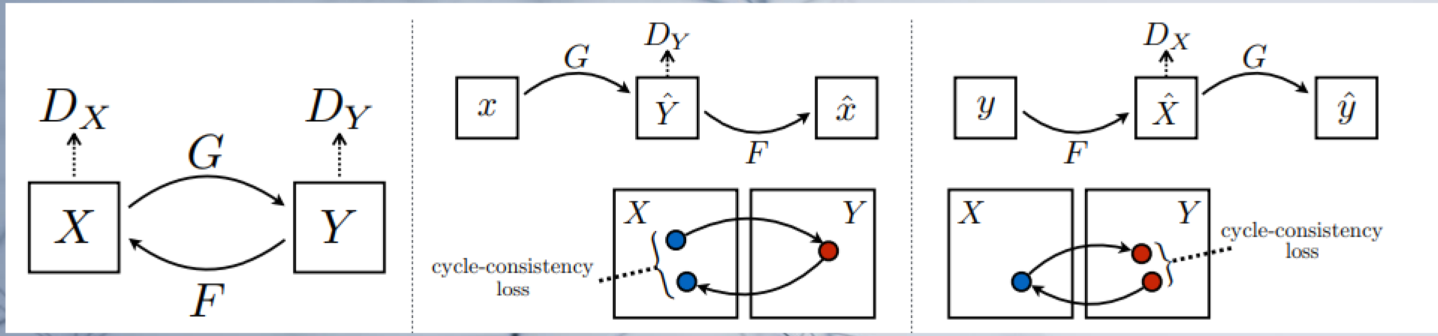
$$\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) + \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) + \lambda \mathcal{L}_{\text{cyc}}(G, F)$$



- $\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y)$: Adversarial loss for generator G and discriminator D_Y, ensuring that images generated by G from domain X look like they belong to domain Y.
- $\mathcal{L}_{\text{GAN}}(F, D_X, Y, X)$: Adversarial loss for generator F and discriminator D_X, ensuring that images generated by F from domain Y look like they belong to domain X.
- $\mathcal{L}_{\text{cyc}}(G, F)$: Cycle consistency loss, ensuring that an image from domain X can be translated to domain Y and back to X (and similarly for Y to X to Y), preserving the original image identity.

Cycle GAN

Loss Functions



The adversarial loss for G and D_Y is defined as:

$$\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{\text{data}}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x)))]$$

Similarly, the adversarial loss for F and D_X is:

$$\mathcal{L}_{\text{GAN}}(F, D_X, Y, X) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D_X(x)] + \mathbb{E}_{y \sim p_{\text{data}}(y)}[\log(1 - D_X(F(y)))]$$

Forward Cycle Consistency:
An image x from domain **X** should be able to be transformed to domain **Y** using **G** then back to domain **X** using **F**, resulting in the original image x

$$\mathcal{L}_{\text{cyc}}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1].$$

Cycle-consistency loss of CycleGAN.

This ensures that **G** and **F** are inverses of each other, preserving the original image through the translation cycle.

Cycle GAN

Loss Functions

$$\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) + \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) + \lambda \mathcal{L}_{\text{cyc}}(G, F)$$

Impact of λ
The value of λ affects how much emphasis is placed on maintaining the cycle consistency during training.

High λ:
- **Focus –** Cycle consistency.
- **Effect –** Better identity preservation, less realistic images.

- **Example**:
  - λ=10 ensures translations are reversible.

Low λ:
- **Focus –** Adversarial loss.
- **Effect –** More realistic images, poor identity preservation.

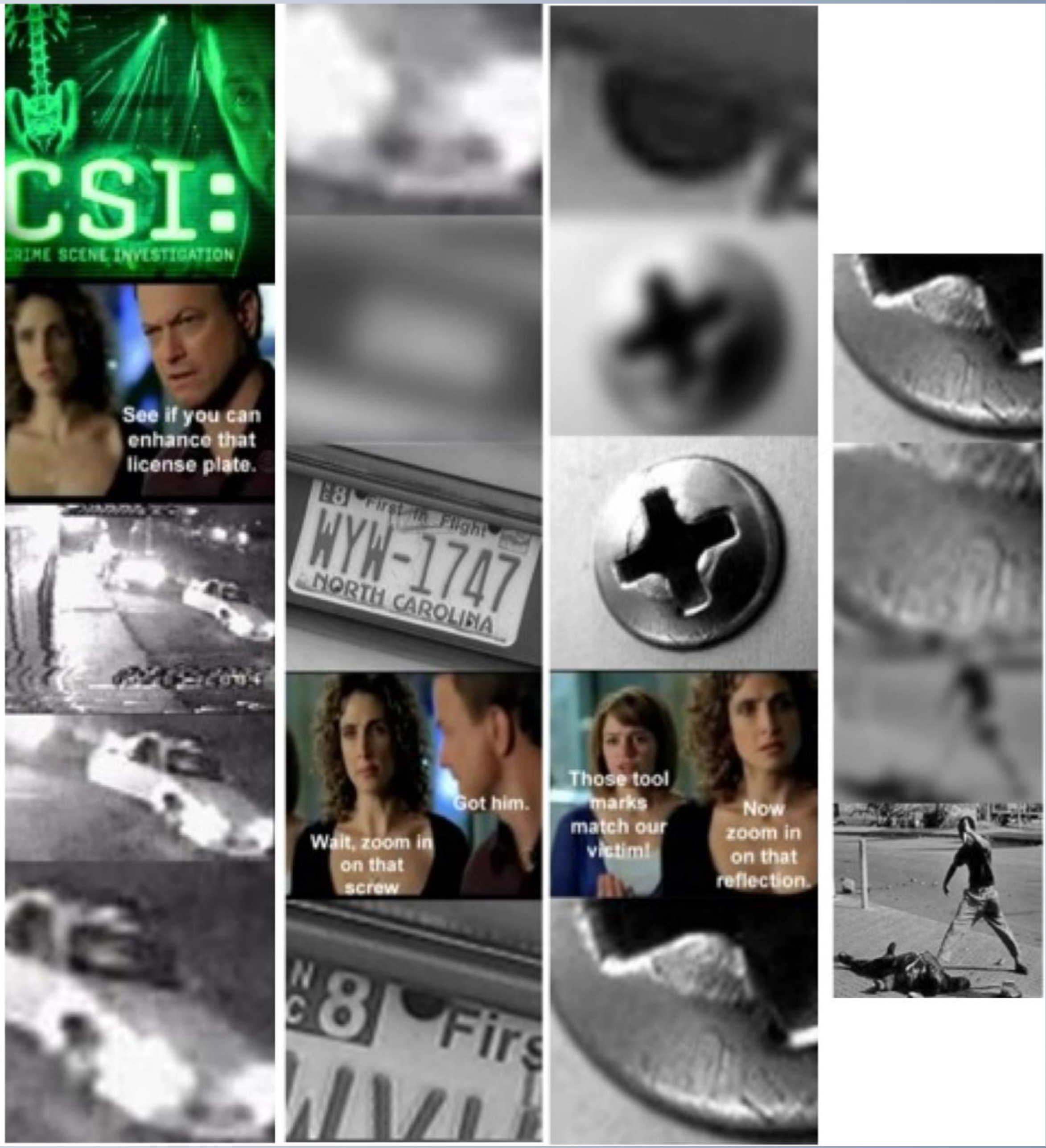
- **Example**:
  - λ=0.1 focuses on realistic target domain images.

Balancing λ is crucial for achieving both realistic and consistent results.

Super resolution

What is Super Resolution?

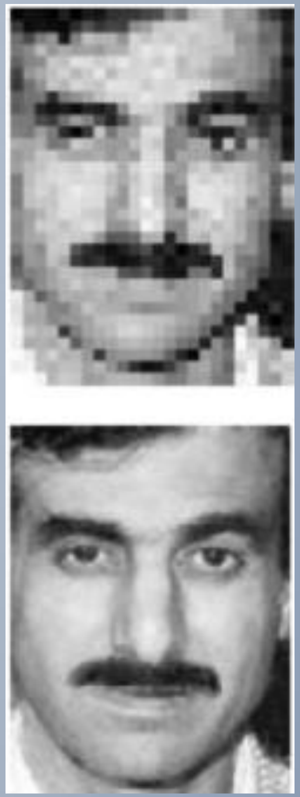
- ❑Enhancing the resolution of images using neural networks
- ❑ Produce a higher quality image
- ❑Detailed and sharper features



Super resolution

Super Resolution and Generative Adversarial Networks (GAN)

- Creates high-resolution images
- Discriminator: Distinguishes real vs. generated images
- Training: Produces realistic, detailed images



Super resolution

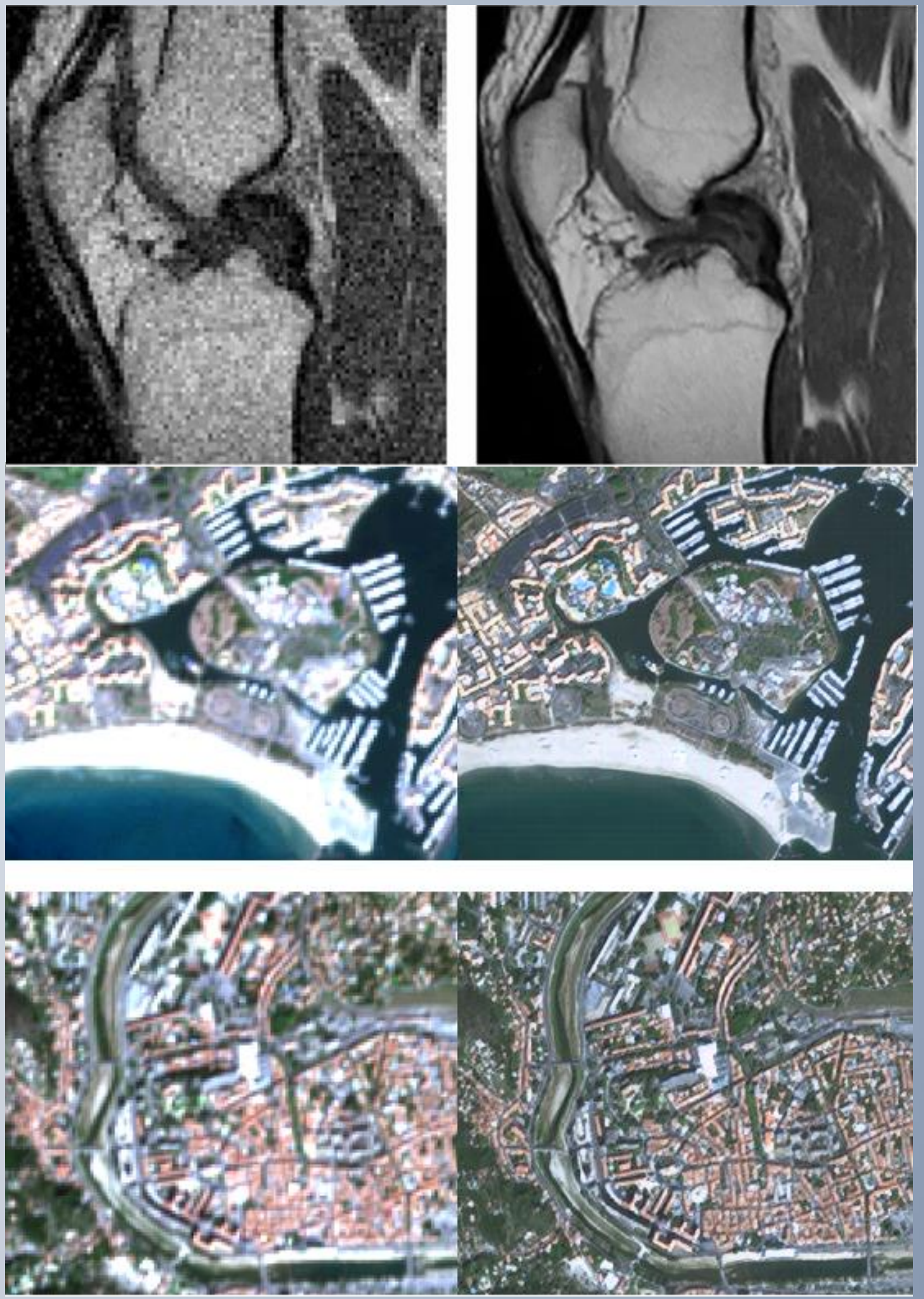
Some Applications of Super Resolution

Medical: Enhancing the resolution of medical scans for better diagnosis.

Satellite Imagery: Improving the detail in satellite photos for research and monitoring.

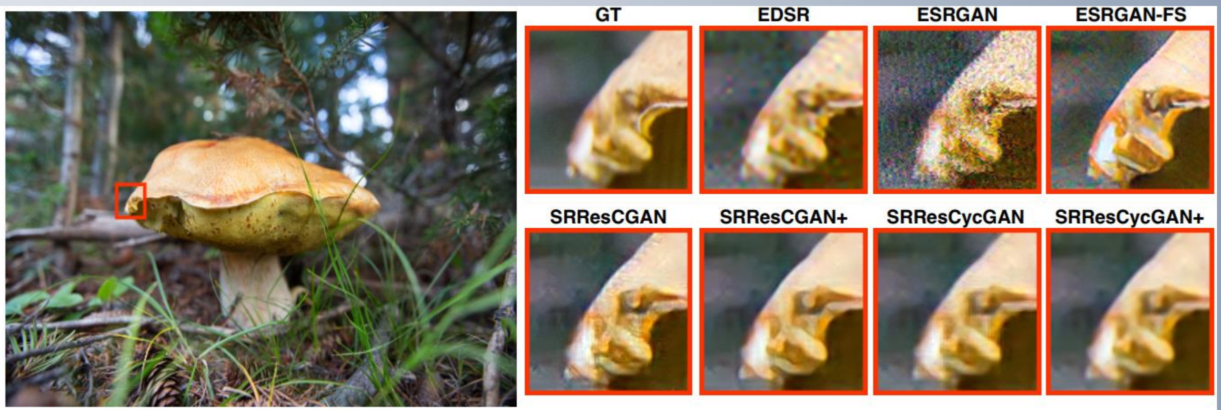
Surveillance: Enhancing video footage for security purposes.

Photography: Improving the quality of photos and videos taken using devices like smartphone and cameras.



Super Resolution GAN Models

Models	Methods	Key words
LAPGAN [32]	CGAN	CGAN with Laplacian Pyramid for image super-resolution
LSMGAN [107]	CGAN	CGAN with local saliency maps for retinal image super-resolution
PGGAN [70]	GAN	Progressive growing GAN for image super-resolution
ESRGAN [148]	SRGAN	SRGAN with Residual-in-Residual Dense Block (RRDB) and relativistic discriminator for image super-resolution
MPDGAN [84]	GAN	GAN with multi-discriminators for image super-resolution
ESRGAN+ [123]	ESRGAN	ESRGAN with Residual-in-Residual Dense Residual Block (RRDRB) for image super-resolution
SPGAN [180]	GAN	GAN with identity-based discriminator for face image super-resolution
MLGE [78]	LAPGAN	LAPGAN with edge information for face image super-resolution
SD-GAN [104]	GAN	GAN for remote sensing image super-resolution
PathSRGAN [103]	SRGAN	SRGAN with RRDB for cytopathology image super-resolution
Enlighten-GAN [48]	GAN	GAN with enlighten block for remote sensing image super-resolution
TWIST-GAN [33]	GAN	GAN with wavelet transform (WT) for remote sensing image super-resolution
SCSE-GAN [112]	GAN	GAN with SCSE block for image super-resolution
MFAGAN [25]	GAN	GAN with multi-scale feature aggregation net for image super-resolution
TGAN [34]	GAN	GAN with visual tracking and attention networks for image super-resolution
DGAN [173]	GAN	GAN with disentangled representation learning and anisotropic BRDF reconstruction for image super-resolution
DMGAN [158]	GAN	GAN with two same generators for image super-resolution
G-GANISR [132]	GAN	GAN with gradual learning for image super-resolution
SRGAN [83]	GAN	GAN with deep ResNet for image super-resolution
RaGAN [69]	GAN	GAN with relativistic discriminator for image super-resolution
LE-GAN [133]	GAN	GAN with a latent encoder for realistic hyperspectral image super-resolution
NCSR [77]	GAN	GAN with a noise conditional layer for image super-resolution
Beby-GAN [89]	GAN	GAN with a region-aware adversarial learning strategy for image super-resolution
MA-GAN [159]	GAN	GAN with pyramidal convolution for image super-resolution
CMRI-CGAN [157]	CGAN	CGAN with optical flow component for magnetic resonance image super-resolution
D-SRGAN [31]	SRGAN	SRGAN for image super-resolution
LMISR-GAN [105]	GAN	GAN with residual channel attention block for medical image super-resolution



'0815' image from DIV2K val-set

SRGAN
Layers architecture

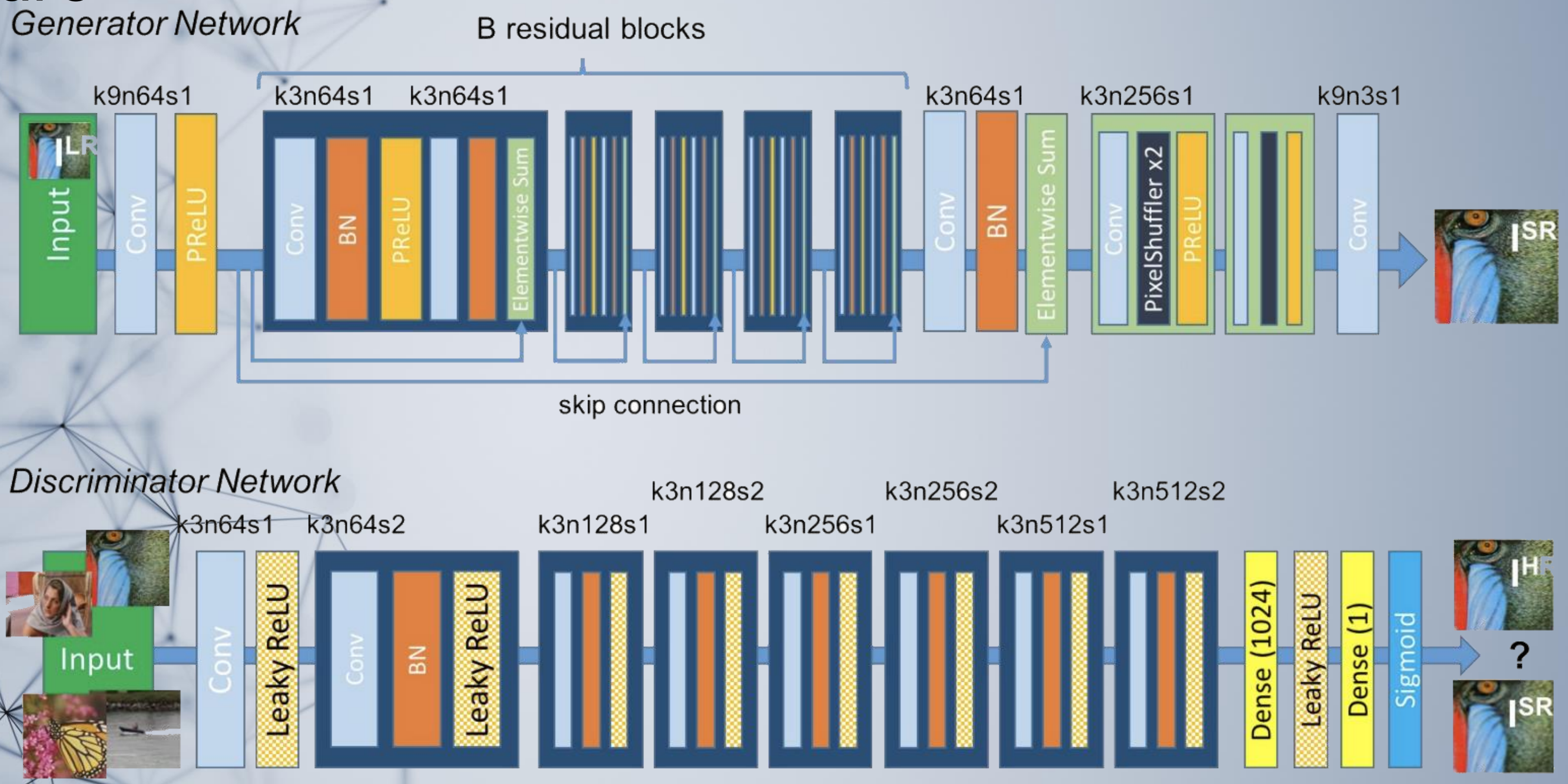
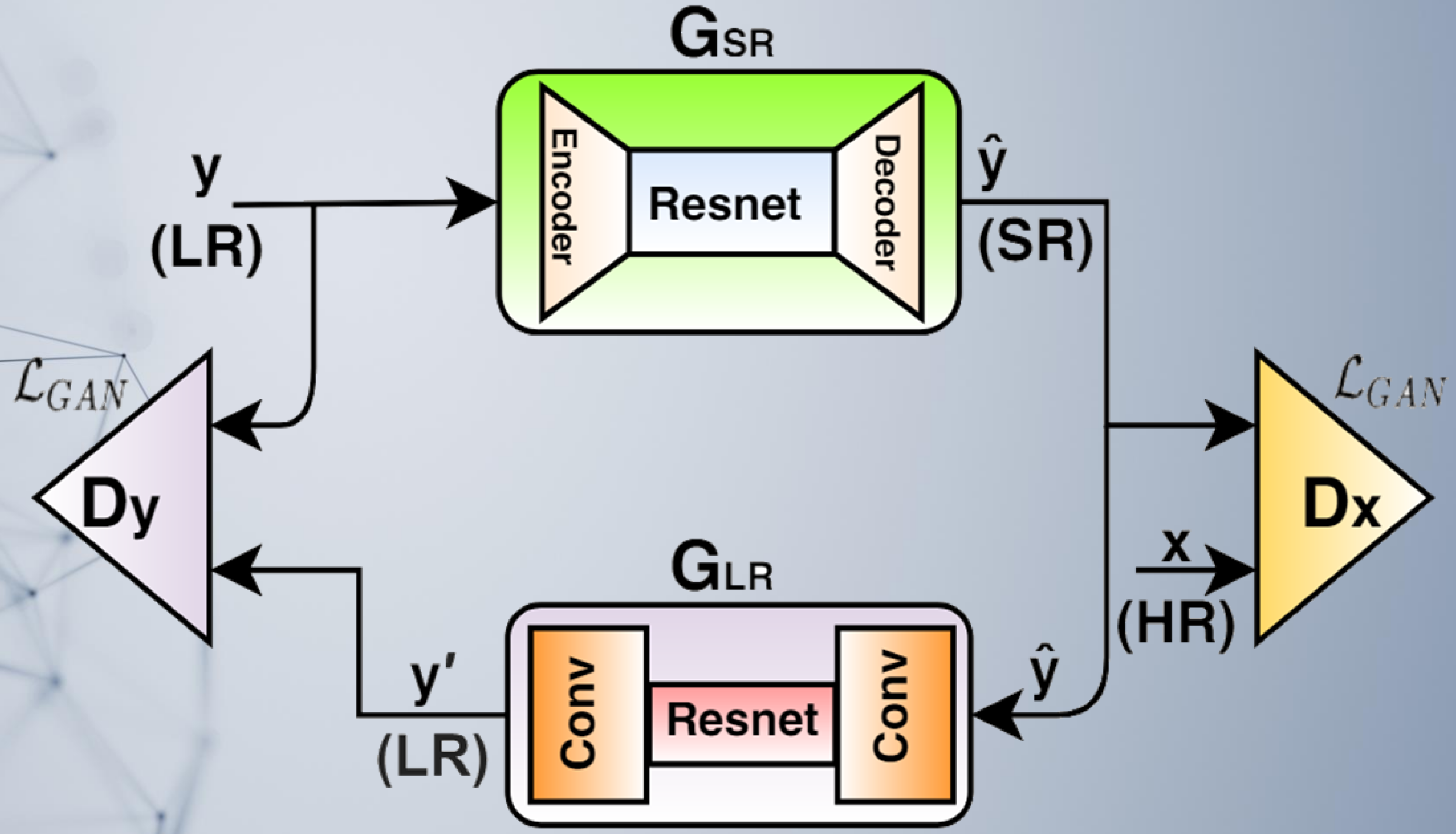


Figure 4: Architecture of Generator and Discriminator Network with corresponding kernel size (k), number of feature maps (n) and stride (s) indicated for each convolutional layer.

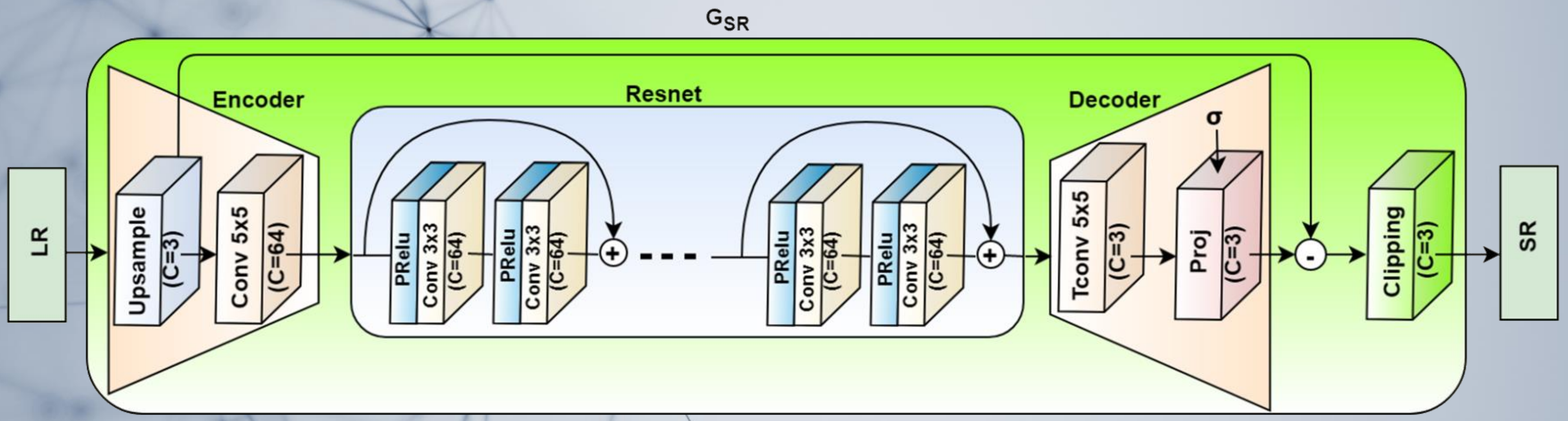
SRResCycGAN

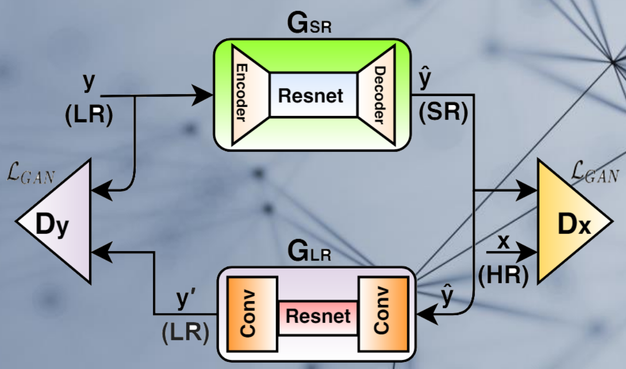
Blocks architecture



SRresCycGAN

Generator(SR) Architecture





Super Resolution Generative Adversarial Network (SRresCycGAN)

PReLU (Parametric Rectified Linear)

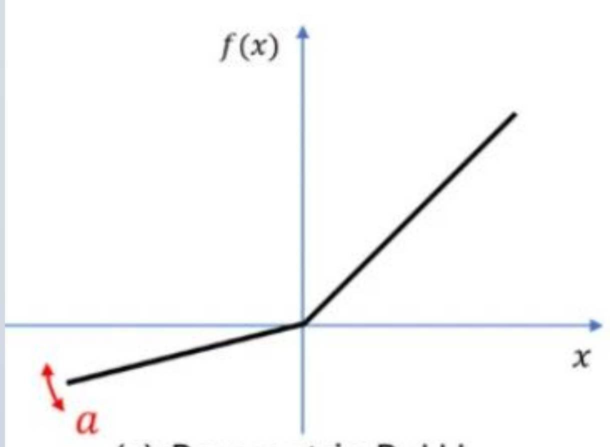
•**Advantages**

- Addresses "dying ReLU"stupni evitagen; rof tneidarg orez-noN :

- Adaptive α: Better performance based on training data

•**Disadvantages**

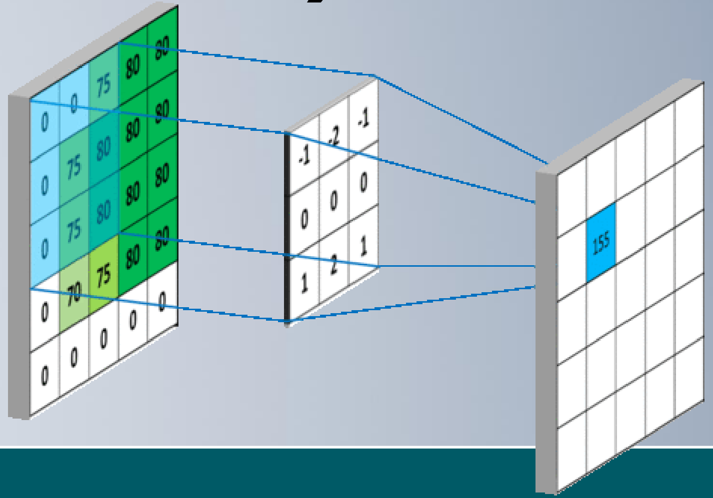
- Additional parametersytixelpmoc ledom sesaercnI :



(c) Parametric ReLU

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha x & \text{if } x \leq 0 \end{cases}$$

Super Resolution Generative Adversarial Network (SRresCycGAN)

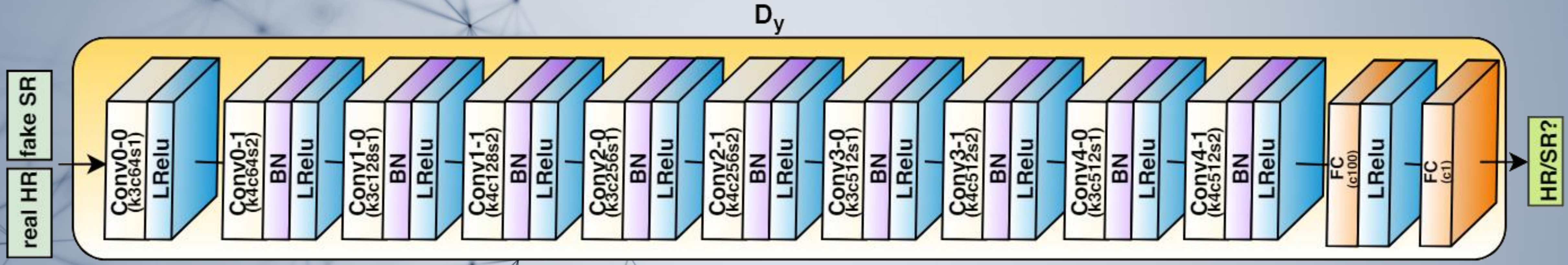


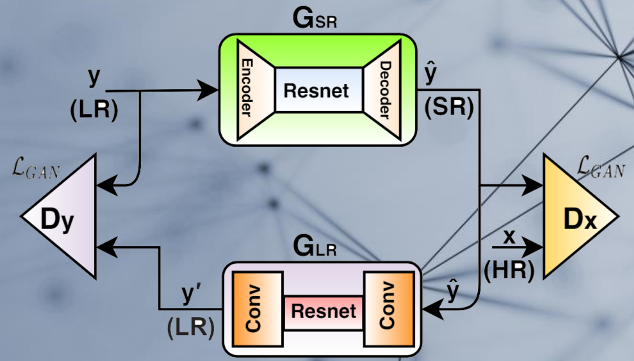
SRResCycGAN Architecture - Convolution

Concept	Explanation	Role
Filters (Kernels)	Small matrices used for detecting features like edges and textures	Creating a feature map that represents detected features in the image
Stride	The step size that the filter takes as it moves across the image	Determining the size of the feature map; larger stride reduces dimensions, smaller stride maintains larger dimensions
Padding	Adding zeros around the original image	Preserving edge features and controlling the size of the output image after convolution
Activation Function	Nonlinear function that adds the ability to learn and generalize complex data	Adding non-linearity to the model; for example, ReLU helps with the vanishing gradient problem

Super Resolution Generative Adversarial Network (SRResCycGAN)

Discriminator(SR) Architecture

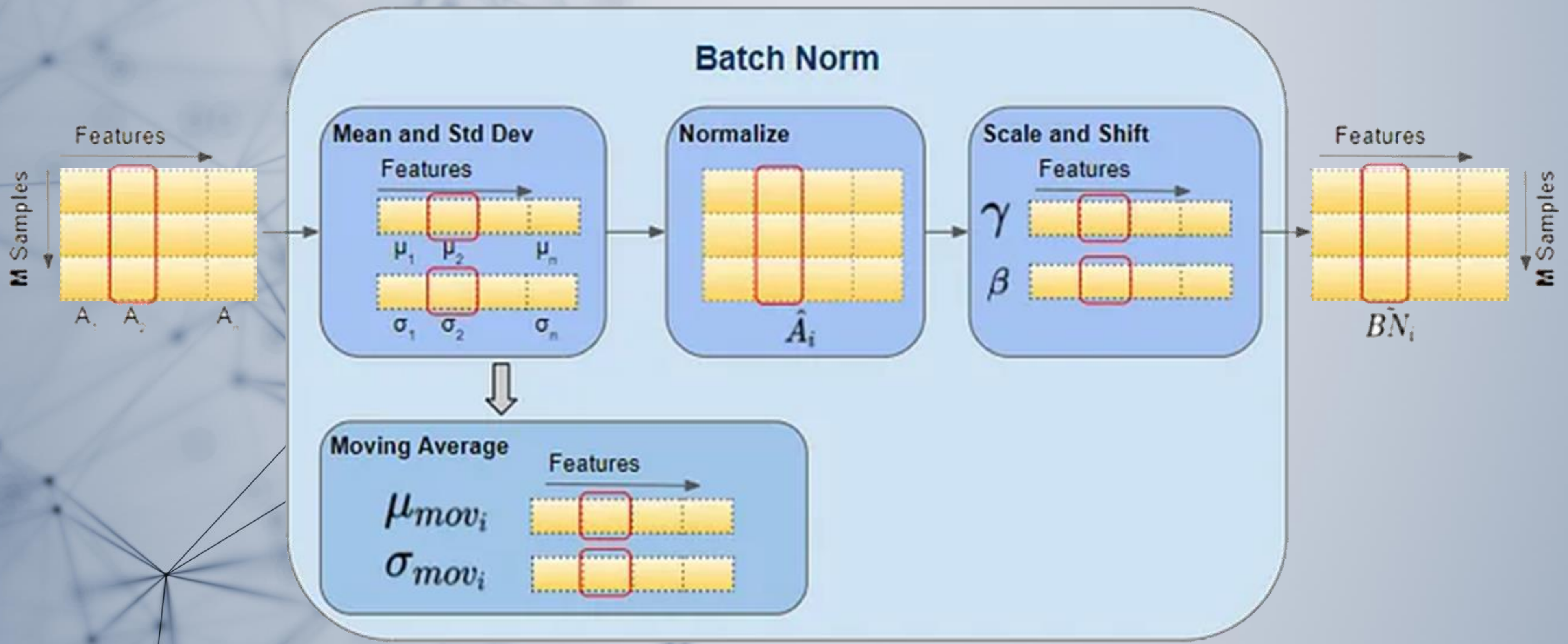




Discriminator(SR)

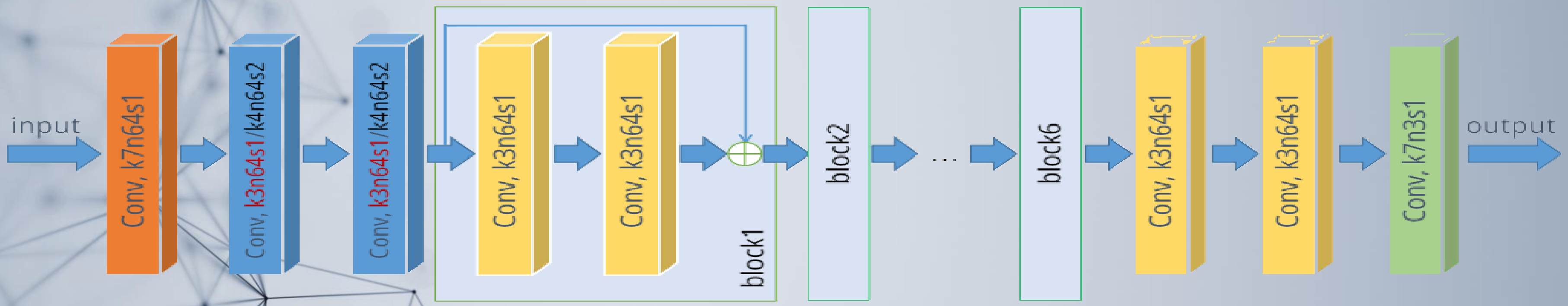
Discriminator Architecture layers- Batch Normalization

It aims to stabilize and speed up the training of neural networks by reducing internal covariate shift

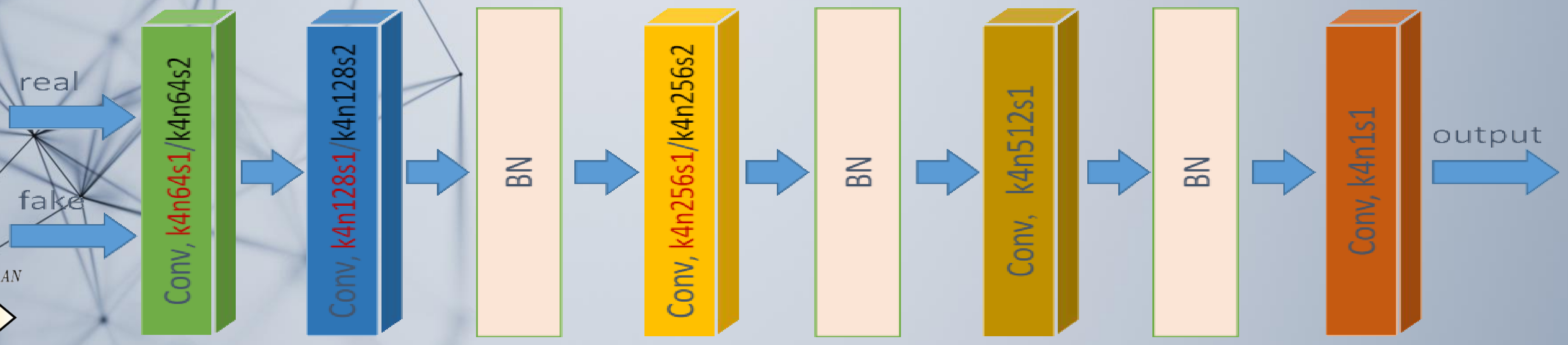


Super Resolution Generative Adversarial Network (SRResCycGAN)

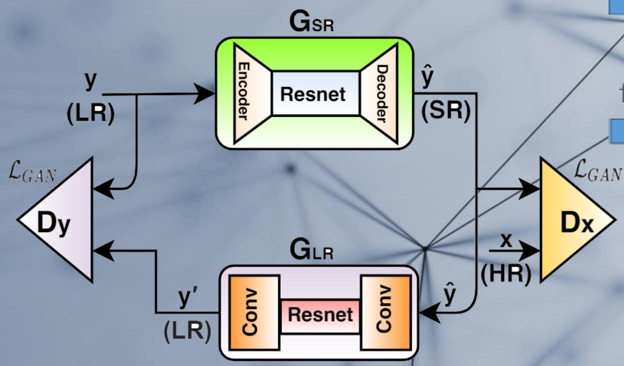
Generator & Discriminator (LR) Architecture



(a) Generator



(b) Discriminator



Our Paper

Deep Generative Adversarial Residual Convolutional Networks for Real-World Super-Resolution

Rao Muhammad Umer Gian Luca Foresti Christian Micheloni
University of Udine, Italy.
engr.raoumer943@gmail.com, {gianluca.foresti, christian.micheloni}@uniud.it

Abstract

Most current deep learning based single image super-resolution (SISR) methods focus on designing deeper / wider models to learn the non-linear mapping between low-resolution (LR) inputs and the high-resolution (HR) outputs from a large number of paired (LR/HR) training data. They usually take as assumption that the LR image is a bicubic down-sampled version of the HR image. However, such degradation process is not available in real-world settings i.e. *inherent sensor noise, stochastic noise, compression artifacts, possible mismatch between image degradation process and camera device. It reduces significantly the performance of current SISR methods due to real-world image corruptions. To address these problems, we propose a deep Super-Resolution Residual Convolutional Generative Adversarial Network (SRResCGAN*[1]*) to follow the real-world degradation settings by adversarial training the model with*

The Problem

- Image enhancement to low resolution images.
- Reconstruct low resolution images from high resolution images.
- (Optional) Add distortion to images and construct high resolution images with denoising.

The Dataset

DIV2K – DIV2K is a popular single-image super-resolution dataset which contains 1,000 images with different scenes and is split to 800 images for training, 100 images for validation and 100 images for testing.

Cat-Dog-Bird – Cat-Dog-Bird is a single-image super-resolution dataset which contains 13,000 images of different types and breeds of Cats, Dogs and Birds. The dataset is split into 3 parts, each part containing the images for either cats, dogs or birds. Each part contains around 4,000 images.

The Experiment From the Article

Objective:

- Develop a super-resolution method that works well on real-world images, addressing the domain gap between low-resolution and high-resolution images.

Training:

- **Data:**
  - 2650 high-resolution images with synthetic degradations (noise, compression artifacts).
  - 800 clean high-resolution images from DIV2K dataset.
  - Additional 19000 low-resolution and high-resolution images from AIM2020 Real Image SR Challenge.

- **Loss Functions:**
  - Perceptual Loss: Ensures perceptual quality.
  - GAN Loss: Focuses on high-frequency details.
  - Content Loss (L1): Maintains content fidelity.
  - Total-Variation Loss: Enhances image sharpness.
  - Cyclic Loss: Ensures consistency between low-resolution and high-resolution images.

- **Optimizers:**
  - Adam optimizer with specific parameters.
  - Learning rate adjustments after specific iterations.

METRICS

Peak Signal-to-Noise Ratio (PSNR)

PSNR is a widely used metric to measure the **quality of reconstructed** images compared to their reference

<u>**where:**</u>

• **MAX** is the **maximum possible** pixel value of the image

• **MSE** is the Mean Squared Error between the **reconstructed and the reference** image.

$$\mathrm{PSNR} = 10 \log_{10}\left(\frac{\mathrm{MAX}^2}{\mathrm{MSE}}\right)$$

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}\left(I_{\mathrm{ref}}(i) - I_{\mathrm{gen}}(i)\right)^2$$

METRICS
Structural Similarity Index (SSIM)

SSIM is a perceptual metric that quantifies image quality degradation caused by processing, such as compression or noise.

Unlike PSNR, SSIM considers changes in structural information, luminance, and contrast. It is defined as:

$$\mathrm{SSIM}(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

where:

- μ_x and μ_y are the mean intensities of images x and y.
- σ_x^2 and σ_y^2 are the variances of x and y.
- σ_{xy} is the covariance between x and y.
- C_1 and C_2 are constants to stabilize the division.

METRICS

Learned Perceptual Image Patch Similarity (LPIPS)

LPIPS is a metric used to measure the perceptual similarity between images, aligning more closely with human visual perception than traditional metrics like PSNR and SSIM.

$$\mathrm{LPIPS}(x, y) = \sum_l \frac{1}{H_l W_l} \sum_{h=1}^{H_l} \sum_{w=1}^{W_l} ||w_l \odot (\hat{y}_l^{hw} - \hat{x}_l^{hw})||_2^2$$

Where:

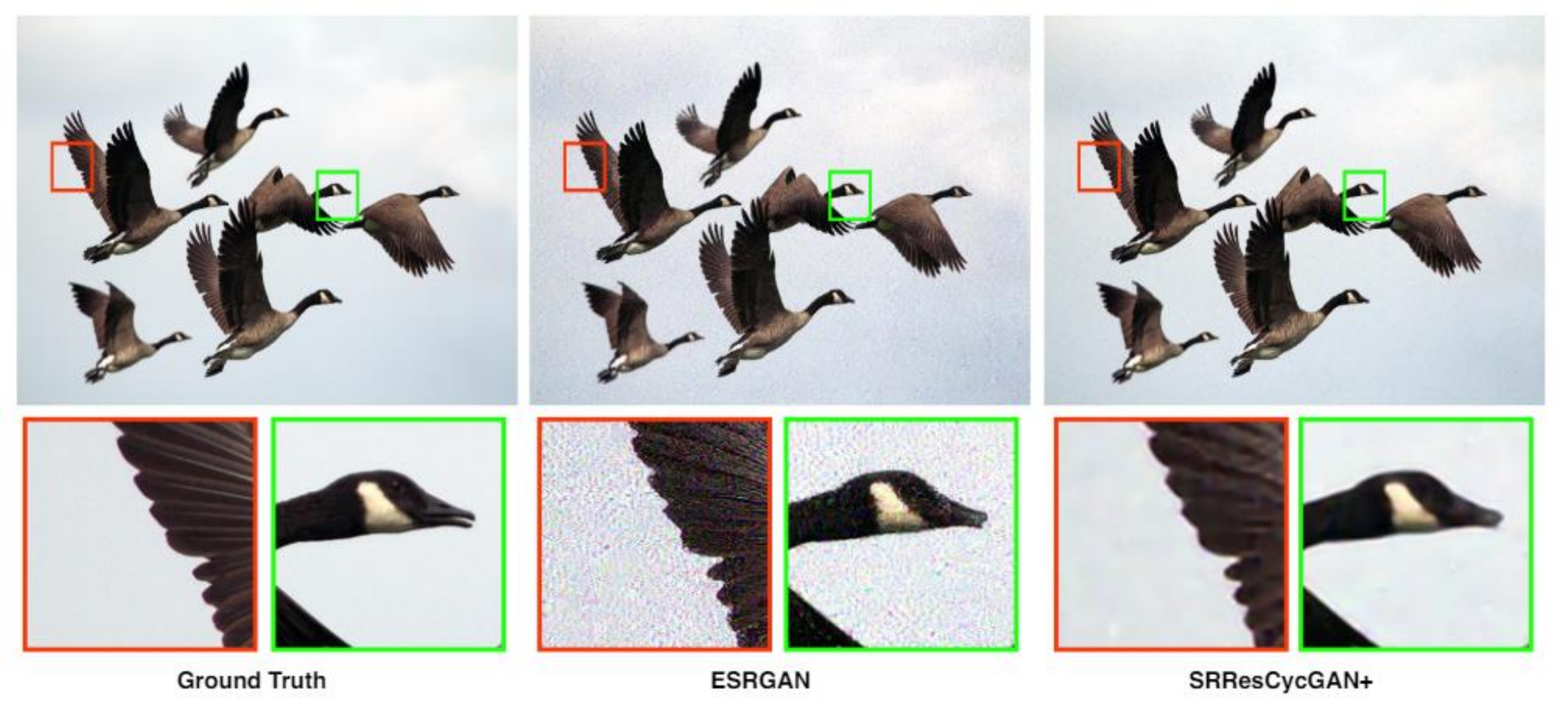
- x and y are the two images being compared.
- l is the layer index in the pretrained network.
- $\hat{x}_l$ and $\hat{y}_l$ are the normalized feature maps of the images at layer l.
- H_l and W_l are the height and width of the feature map at layer l.
- w_l is a learned weight vector that adjusts the contribution of each feature channel.
- $\odot$ denotes element-wise multiplication.

Model Performance

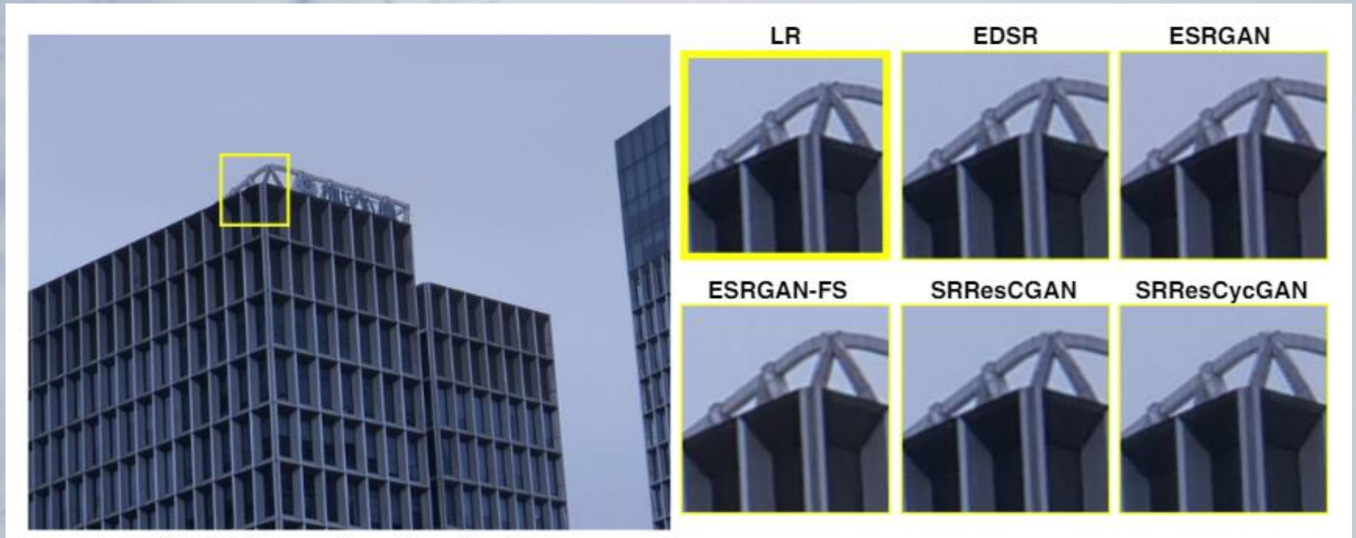
SR method	Cyclic Path	Network structure	PSNR↑	SSIM↑	LPIPS↓
SRResCycGAN	×	$\mathbf{y} \rightarrow \mathbf{G_{SR}} \rightarrow \hat{\mathbf{y}}$	25.05	0.67	**0.3357**
SRResCycGAN	✓	$\mathbf{y} \rightarrow \mathbf{G_{SR}} \rightarrow \hat{\mathbf{y}} \rightarrow \mathbf{G_{LR}} \rightarrow \mathbf{y}'$	26.13	0.71	0.3911
SRResCycGAN+	✓	$\mathbf{y} \rightarrow \mathbf{G_{SR}} \rightarrow \hat{\mathbf{y}} \rightarrow \mathbf{G_{LR}} \rightarrow \mathbf{y}'$	**26.39**	**0.73**	0.4245

		sensor noise ($\sigma = 8$)			compression artifacts ($q = 30$)		
SR methods	#Params	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
EDSR [13]	43M	24.48	0.53	0.6800	23.75	0.62	0.5400
ESRGAN [22]	16.7M	17.39	0.19	0.9400	22.43	0.58	0.5300
ESRGAN-FT [15]	16.7M	22.42	0.55	0.3645	22.80	0.57	0.3729
ESRGAN-FS [5]	16.7M	22.52	0.52	0.3300	20.39	0.50	0.4200
SRResCGAN [18]	380K	25.46	0.67	0.3604	23.34	0.59	0.4431
SRResCycGAN (ours)	380K	25.98	0.70	0.4167	23.96	0.63	0.4841
SRResCycGAN+ (ours)	380K	26.27	0.72	0.4542	24.05	0.64	0.5192
		unknown corruptions [17]					
SRResCGAN [18]	380K	25.05	0.67	0.3357			
SRResCycGAN (ours)	380K	26.13	0.71	0.3911			
SRResCycGAN+ (ours)	380K	26.39	0.73	0.4245			
		real image corruptions [24]					
SRResCycGAN (ours, valset)	380K	28.6239	0.8250	-			
SRResCycGAN (ours, testset)	380K	28.6185	0.8314	-			

Model Results



Model Results



'LR_015' image from the val-set

'LR_050' image from the testset

The Experiment From the Article

Conclusions:

Advantages:

- Effective for real-world super-resolution.
- Suitable for deployment on mobile/embedded devices due to low parameter count.

Performance:

- Outperformed existing methods in both quantitative metrics and quality.

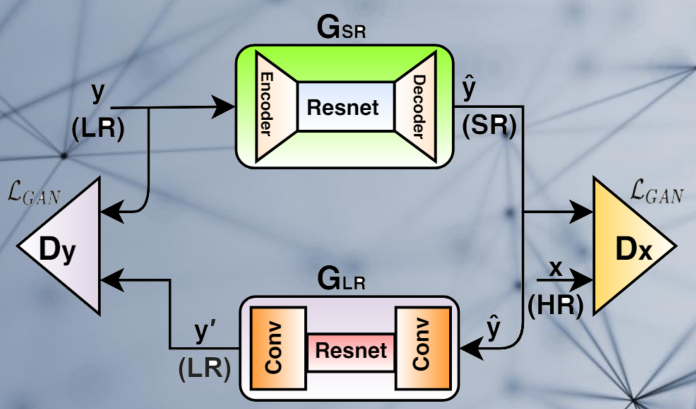
Our Plan

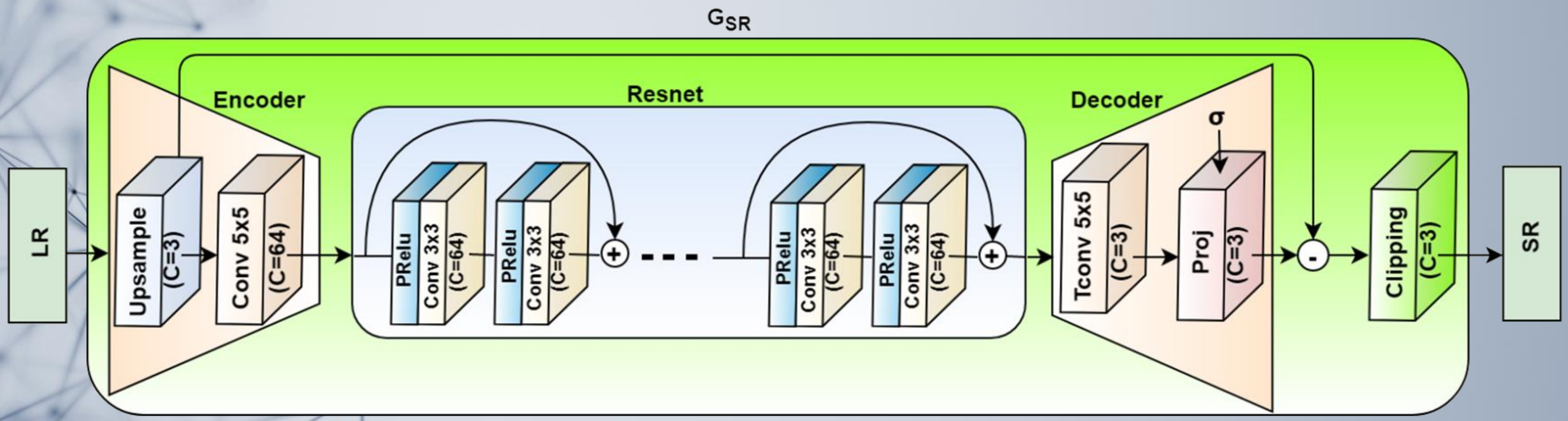
1. Environment Setup
2. Data Collection
3. Data Preprocessing
4. Build the model
5. Train & test the model
6. Fine Tuning
7. Evaluation and Conclusions

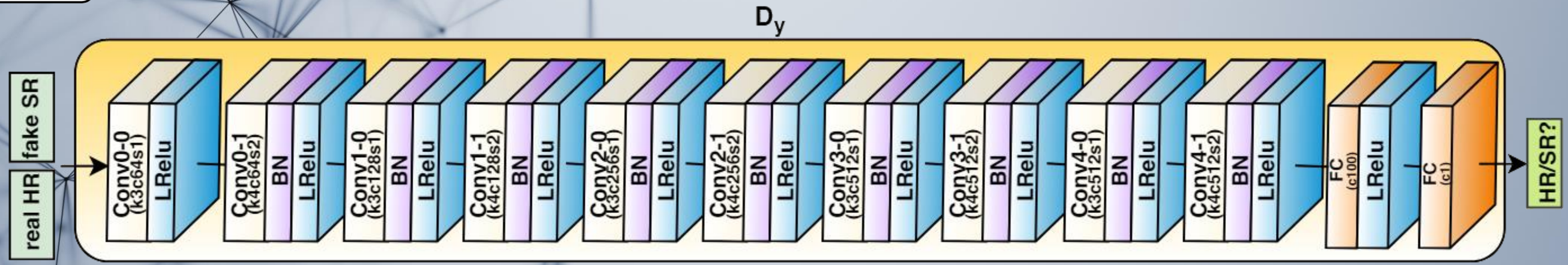
Our Solution

Use SRResCycGAN model to produce High-Resolution images from low resolution images.

Feed the result back to a second generator to generate Low-Resolution images from the high-resolution output, comparing between ground truth and the final low-resolution result.







Thank You!

Only together we shall win!